

Designed by slidesgo / Freepik



GERENCIAMENTO DE CIDADES

Fundamentos do Spring Boot e Bootstrap

Visão Geral

Spring Initializr

Explorando

Bootstrap

Servindo a Página

Verificação de Aprendizado

VISÃO GERAL

Nessa primeira aula prática, daremos início a um projeto do tipo CRUD. Um CRUD é um aplicativo que permite criar (*create*), ler (*read*), alterar (*update*) e excluir (*delete*) dados. Como referência, vamos usar os dados de cidades. Uma cidade está associada a um único estado. Portanto, para criar uma cidade precisamos informar o nome da cidade e o estado onde ela se encontra.

CREANDO UM PROJETO SPRING BOOT

O primeiro passo para criar um novo projeto usando o Spring Boot é visitar o [Spring Initializr](#). O [Spring Initializr](#) permite a definição das características essenciais para o projeto, como a versão do Java que será usada e os dados do projeto. Além disso, ele permite selecionar as dependências quer serão necessárias no projeto.

The screenshot shows the Spring Initializr interface. On the left, under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. In the 'Dependencies' section, 'Spring Web' is checked. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

Na parte superior do lado direito, a tela permite selecionar o tipo de projeto que será usado. Conforme definido na semana anterior, vamos usar o Maven para gerenciar as dependências do projeto. Do lado direito dessa opção, é possível escolher a linguagem que será usada. Apesar da JVM permitir outras linguagens além do Java, nessa disciplina nosso foco é Java.

The screenshot shows the Spring Initializr interface. On the left, under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. In the 'Dependencies' section, 'Spring Web' is checked. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

O Spring Boot é atualizado constantemente. No momento em que essa aula está sendo preparada, a versão estável mais atual é 2.4.3. Essa é a versão que usaremos nesse projeto.

The screenshot shows the Spring Initializr interface. On the left, under 'Project', 'Maven Project' is selected. Under 'Language', 'Java' is selected. In the 'Dependencies' section, 'Spring Web' is checked. At the bottom, there are buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

Em seguida, é preciso definir os meta-dados do projeto. Essas informações são usadas pelo Maven para, por exemplo, registrar o projeto no repositório do Maven. Também é preciso selecionar o tipo de empacotamento que será usado. No passado, aplicativos Web em Java costumavam ser empacotados em formato war. Contudo, hoje em dia o padrão é jar. Por isso, esse é o formato que iremos utilizar. Por fim, neste projeto vamos começar utilizando o Java 11, apesar do Java 16 ter sido lançado recentemente no momento em que essa aula está sendo preparada.

The screenshot shows the 'Project Metadata' section of the Spring Initializr interface. It includes fields for Group (br.edu.utfpr.esqjava), Artifact (crud-cidades), Name (crud-cidades), Description (Um CRUD de Cidades para demonstrar as funcionalidades do Spring Boot), Package name (br.edu.utfpr.esqjava.crudcidades), Packaging (jar), and Java version (11).

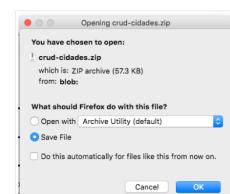
Um projeto de software consiste de uma ou mais dependências. Essas dependências são bibliotecas externas ou do próprio framework que são usadas para facilitar o desenvolvimento. O [Spring Initializr](#) permite a seleção de algumas dependências comuns em projetos com Spring Boot. Para selecionar as dependências necessárias para o projeto, basta clicar no botão **ADD DEPENDENCIES**, no lado direito da tela.

The screenshot shows the 'Dependencies' section of the Spring Initializr interface. It lists several categories of dependencies: DEVELOPER TOOLS (Spring Boot DevTools), WEB (Spring Web, Spring Reactive Web), REST Repositories, Spring Session, Rest Repositories HAL Explorer, and Spring HATEOAS. A search bar at the top allows for filtering dependencies.

Para finalizar a criação do projeto, basta clicar no botão **GENERATE**, na parte inferior da tela. O [Spring Initializr](#) irá gerar um projeto Maven com as especificações informadas e dependências selecionadas.

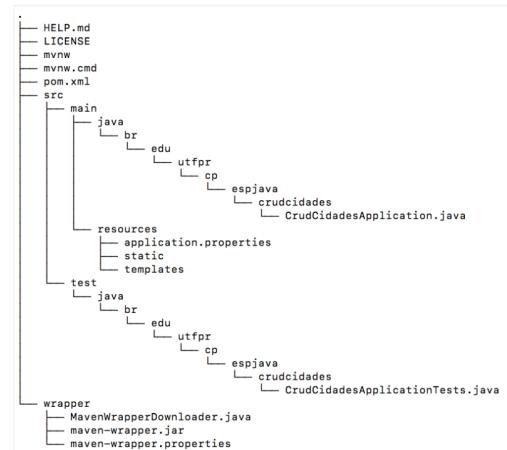
The screenshot shows the 'GENERATE' section of the Spring Initializr interface. It includes buttons for 'GENERATE', 'EXPLORE', and 'SHARE...'.

Em seguida, o navegador deve solicitar o download de um arquivo zip. Esse arquivo corresponde ao projeto comprimido. Salve o arquivo na pasta da sua preferência. Em seguida, descomprima o arquivo e abra o projeto no seu editor preferido.



EXPLORANDO O PROJETO CRIADO

A Figura abaixo mostra a estrutura do projeto Maven criado pelo *Spring Initializer*. Na raiz, o arquivo mais importante é o `pom.xml`. É esse arquivo que armazena as configurações do Maven, o que inclui meta-dados e dependências, entre outros. Dentro da pasta `src`, existem duas pastas: `main` e `test`. Enquanto a pasta `main` é usada para os arquivos-fonte, a pasta `test` é usada para os arquivos de teste, como testes unitários, por exemplo.



A pasta `main` se divide em `java` e `resources`. Enquanto a pasta `java` guarda as classes Java usadas no projeto, a pasta `resources` guarda os arquivos de configuração (`application.properties`), além de outras duas pastas. A pasta `static` é usada para armazenar conteúdo estático, como páginas `html` estáticas, enquanto a pasta `templates` é usada para armazenar conteúdo dinâmico, como páginas `html` que são alteradas dinamicamente durante a execução da aplicação.

Vocês vão perceber também que o projeto já vem com duas classes Java: uma dentro da pasta `main` e outra na pasta `test`. A classe Java dentro da pasta `main` contém o método que inicializa a aplicação, enquanto a classe Java dentro da pasta `test` é um teste unitário de exemplo.

Corn o projeto aberto no seu editor ou IDE de preferencia, vamos explorar a classe `CrudCidadesApplication.java`. Como vocês podem perceber, é uma classe Java comum. Ela tem o método `main`, que também é comum em aplicações Java. Esse método é o primeiro método a ser executado quando a aplicação é inicializada.

O código dentro do método `main`, na linha 10, e anotação `@SpringBootApplication`, na linha 6, são usados pelo Spring Boot para definir que esse é um projeto Spring Boot e inicializar as configurações padrão para esse tipo de projeto.

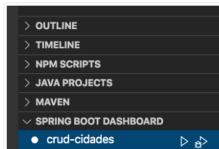
Este projeto está disponível no [Github](#), na branch `projeto-vazio`.

```

1 package br.edu.utfpr.cp.espjava.crudcidades;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class CrudCidadesApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(CrudCidadesApplication.class, args);
11     }
12 }
13

```

Se você está usando o VS Code com os plug-ins para Java e Spring Boot recomendados na aula anterior, você pode executar a aplicação clicando no botão `play`, na janela `SPRING BOOT DASHBOARD`.



Ao entrar em execução, você vai notar a janela de controle que, normalmente, é mostrada no canto superior direito da janela. O botão `STOP` (quadradinho vermelho) é usado para parar a aplicação.



O painel inferior do editor mostra o log de execução do Spring Boot. A penúltima mensagem, `Tomcat started on port(s): 8080 (http) with context path ''`, mostra que a aplicação está em execução e pode ser acessada na porta 8080. Se você estiver usando uma IDE, como Netbeans ou Eclipse, o procedimento de execução não é muito diferente. Basta localizar o botão `play` e dar início à execução. Se você for um adepto da linha de comando, você pode iniciarizar o aplicativo entrando na pasta do projeto pelo seu terminal e digitando `mvn spring-boot:run`. Observe que você precisa ter seu ambiente configurado, com Java e Maven instalados.

É importante ressaltar que você deve usar a IDE ou editor que você tiver mais familiaridade para ter produtividade neste curso. Se você já tiver mais habilidade com tecnologia e quiser testar algo novo, então fique à vontade para se aventurar com o VS Code. Nada nesse curso é específico do VS Code, por isso, todo o conteúdo pode ser feito no editor/IDE da sua preferência.



Na sua máquina local, basta abrir o navegador e digitar: `http://localhost:8080`. Contudo, observe que a aplicação ainda não faz nada. Por isso, tudo que você vai conseguir é uma mensagem de `erro 404`. Não se preocupe, agora vamos começar a moldar a aplicação da forma como queremos.

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Mar 02 16:22:04 BRT 2021
There was an unexpected error (type=Not Found, status=404).

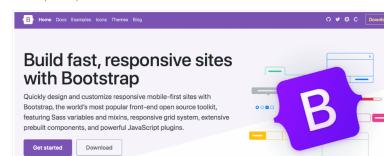
criando uma página estática com Bootstrap
Se o aplicativo ainda não está em execução, pare a execução antes de continuar. Se o aplicativo foi iniciado em linha de comando, use `CTRL + C` para parar a execução.

Como primeiro projeto, vamos criar um CRUD de cidades. Um CRUD é a sigla usada para representar um aplicativo capaz de criar (criar), ler (read), alterar (update) e excluir (delete) dados. Nesse caso, estamos usando a representação de uma cidade, que sempre pertence a um único estado. Esse exemplo é simples e familiar para todos. Para isso, precisamos de um protótipo de como será a página principal. Eu usei o draw.io para criar o protótipo na Figura a seguir.



A página mostra um rótulo no topo, identificando o projeto. Logo em seguida, um formulário é usado para entrada de dados. Vamos usar apenas dois dados: nome e estado. É assim mesmo, esse primeiro projeto é para ser simples. Logo abaixo do formulário, vêm uma tabela que lista as cidades já criadas. Para cada cidade listada na tabela, é possível alterar os dados, ou excluir a cidade, através de botões de interação.

Para implementar o protótipo criado na Figura anterior, é necessário adicionar HTML no arquivo `/resources/static/crud.html`. Esse arquivo será usado para criar a página HTML, que é a versão web do aplicativo. Uma vez que o Java consiste de HTML, CSS e, possivelmente, JavaScript, além de classes Java. Por isso, vamos sim usar HTML e CSS mesmo, sendo uma aplicação Web Java. É claro que você pode criar todo seu CSS, formate suas páginas de forma que achar melhor. Contudo, nesse curso nós vamos usar o Bootstrap 4. O Bootstrap é uma biblioteca que traz muitas opções de formatação à priori. Com o Bootstrap você consegue formatar componentes HTML de forma rápida e simples.



Eu não sou um expert em CSS nem em Bootstrap. Sério, falo isso sem medo. Não só precisamos saber tudo, mas precisamos conhecer quem é. Por isso, eu preferencialmente uso a referência do W3Schools, além da documentação do próprio Bootstrap quando preciso otimizar o projeto. Lembrando que esse NÃO é um curso de Bootstrap. Por isso, vamos usar apenas o que precisamos para montar nossa página.



A primeira coisa que precisamos fazer é adicionar conteúdo HTML ao arquivo `/resources/static/crud.html`. Algo como a figura abaixo:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta http-equiv="X-UA-Compatible" content="IE=edge">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>CRUD Cidades</title>
8   </head>
9   <body>
10
11 </body>
12 </html>
```

Em seguida, precisamos tornar o Bootstrap disponível na nossa aplicação. Uma forma simples de fazer isso é usando CDN. O CDN é uma rede de servidores que dá acesso aos arquivos de configuração do Bootstrap. Basta adicionar o código da linha 9, conforme a figura a seguir:

```
1 <head>
2   <meta charset="UTF-8">
3   <meta http-equiv="X-UA-Compatible" content="IE=edge">
4   <meta name="viewport" content="width=device-width, initial-scale=1.0">
5   <title>CRUD Cidades</title>
6
7   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" />
8 </head>
```

O Bootstrap usa uma estrutura chamada `container` para definir o layout principal de uma página. Para isso, vamos criar uma seção principal usando um `<div>` aplicando o formato `container-fluid`, conforme mostram as linhas 14-16, na figura a seguir.

```
13 <body>
14   <div class="container-fluid">
15
16   </div>
17 </body>
```

Vamos usar um componente do Bootstrap chamado `Jumbotron` para criar o rótulo superior que identifica o aplicativo, conforme mostra as linhas 18-19. Note que também adicionamos um espaço na margem superior (`mt-3`).

```
18 <body>
19   <div class="container-fluid">
20     <div class="jumbotron mt-3">
21       <h1>GERENCIAMENTO DE CIDADES</h1>
22       <p>LIM CRUD PARA CRIAR, ALTERAR, EXCLUIR E LISTAR CIDADES.</p>
23     </div>
24   </div>
25 </body>
```

Agora, o próximo passo é adicionar o `formulário`. Para isso, vamos criar um formulário logo abaixo do `jumbotren`.

```
26 <body>
27   <div class="container-fluid">
28     <div class="form-group">
29       <label for="Nome">Nome:</label>
30       <input type="text" class="form-control" placeholder="Informe o nome da cidade" id="Nome">
31     </div>
32     <div class="form-group">
33       <label for="Estado">Estado:</label>
34       <input type="text" class="form-control" placeholder="Informe o estado ao qual a cidade pertence" id="Estado">
35     </div>
36     <div>
37       <button type="submit" class="btn btn-primary">Criar</button>
38     </div>
39   </div>
```

O último passo é adicionar uma tabela, logo abaixo do formulário. A tabela deve listar as cidades já existentes. Além disso, é possível adicionar ou excluir uma cidade clicando nos botões correspondentes, apresentados na tabela. Observe que adicionamos vários componentes de formatação para tornar a tabela mais agradável:

```
40 <table class="table table-striped table-bordered table-hover" style="width:100%">
41   <thead class="thead-dark">
42     <tr>
43       <th>Cidade:</th>
44       <th>Nome:</th>
45       <th>Estado:</th>
46       <th>Ações:</th>
47     </tr>
48   </thead>
49   <tbody>
50     <tr>
51       <td>Londrina</td>
52       <td>Paraná</td>
53       <td><a href="#">Alterar</a></td>
54       <td><a href="#">Excluir</a></td>
55     </tr>
56   </tbody>
57 </table>
```

A página está pronta e está bem similar ao protótipo definido. Você pode salvar o arquivo e abrir no seu navegador.

Este projeto está disponível no [GitHub](#), na branch `projeto-pagina-estatica`.



CRIANDO O ACESSO À PÁGINA

Agora que nossa página estática está pronta, precisamos criar uma classe Java que será responsável por receber as solicitações do navegador e enviar a página para o navegador. Para isso, vamos criar um pacote que vamos chamar de `visao`. No VS Code, basta criar uma pasta dentro da pasta `crudcidades`. Em seguida, crie a classe `CidadeController.java`, dentro da pasta `visao`. Se estiver usando outra IDE/editor, siga o procedimento adequado na sua IDE/editor para criar a classe o pacote.

Um pacote em Java é representado na estrutura de diretórios como uma pasta.



Na classe `CidadeController`, crie um método que retorne o nome da página estática como uma `String`, conforme a Figura abaixo.

```
6  public String listar() {
7      return "crud.html";
8 }
```

Por fim, adicione as anotações `@Controller` imediatamente antes da declaração da classe, e `@GetMapping` imediatamente antes da declaração do método.

O VS Code também completa o código pra você, assim como as IDEs. Basta começar a escrever e ele já faz as sugestões. O VS Code também importa as classes usadas pelas bibliotecas referenciadas no `pom.xml`.

```
6 @Controller
7 public class CidadeController {
8
9     @GetMapping("/")
10    public String listar() {
11        return "crud.html";
12    }
}
```

A anotação na linha 6 indica pro Spring Boot que essa classe é uma classe que recebe requisições vindas do navegador. Já a anotação na linha 9 indica pro Spring Boot que qualquer solicitação feita para o endereço indicado, nesse caso, a raiz da URL, deve ser entregue para o método `listar()`. Quando o método retorna o nome da página estática, o Spring Boot entende que ele deve carregar essa página. Execute sua aplicação de dentro da sua IDE/editor ou usando `mvn spring-boot:run` em linha de comando. O resultado deve ser parecido com o da Figura a seguir.

Nome	Estado	Ações
Londrina	Paraná	ALTERAR EXCLUIR

O código desse projeto está disponível no [Github](#), na branch `projeto-serviço-estática`.

Claro, essa página ainda não faz nada. Contudo, esses são os princípios fundamentais para criar uma aplicação Web com Java e Spring Boot. Agora, você pode comparar o código que você desenvolveu nessa aula com o código do [Petclinic](#), apresentado na aula anterior. Na nossa próxima aula, vamos evoluir esse código adicionando dinamismo na página e criando um CRUD completo.